An Interface for Creating and Manipulating Curves using a High Degree-of-Freedom Curve Input Device

Tovi Grossman, Ravin Balakrishnan, Karan Singh
Department of Computer Science
University of Toronto
tovi | ravin | karan @dgp.toronto.edu
www.dgp.toronto.edu

ABSTRACT
Current interfaces for manipulating curves typically use a standard point cursor to indirectly adjust curve parameters. We present an interface for far more direct manipulation of curves using a specialized high degree-of-freedom curve input device, called *ShapeTape*. This device allows us to directly control the shape and position of a virtual curve widget. We describe the design and implementation of a variety of interaction techniques that use this curve widget to create and manipulate other virtual curves in 2D and 3D space. The input device is also used to sense a set of user gestures for invoking commands and tools. The result is an effective alternate user interface for curve manipulation that can be used in 2D and 3D graphics applications.

Categories & Subject Descriptors: H.5.2 [Information Interfaces and Presentation]: User Interfaces; I.3.6 [Computer Graphics]: Methodology and Techniques
General Terms: Human Factors; Design
Keywords: High degree-of-freedom input; Curve editing

INTRODUCTION
In computer graphics, curves are a fundamental primitive used in a variety of applications. These include direct use in 2D drawings, specification of surfaces in 3D geometric modeling, and specification of motion and camera paths in animation and virtual reality systems. Thus, the ability to quickly and precisely create an appropriate set of curves is a crucially important task in these graphics applications.

Most current interactive curve manipulation techniques require that the user, to some extent, understand and work with the underlying mathematical representations of curves in order to control its shape and size. Recognizing the limitations of these existing tools, researchers have been working on tools for sketching and refining curves in a more direct manner [2, 4, 5, 8, 9, 11, 15-17]. While sketching is a very useful paradigm for creating and refining curves, another paradigm that has not received as much attention is the use of high degree-of-freedom input devices to directly manipulate virtual curves. In the design industry, however, traditional high degree-of-freedom physical techniques for manipulating curves in clay modeling and paper drawings are still very popular. Here, curves are created directly by copying segments from physical templates (e.g., French curve templates) or using physical tools, which flex to produce curves (e.g., spring steels). Many of these physical techniques allow for very fast and accurate specification of curves, while current virtual techniques are typically more cumbersome. Given the success of these techniques in the real world, it is reasonable to expect that virtual interaction techniques could benefit from the use of physical artifacts more closely matched to the task [7, 13, 16].

In [3], based on physical techniques used in the design industry, Balakrishnan et. al. explored using a high degree-of-freedom curve input device to directly create curves and surfaces. While theirs was the first such system that exploited the affordances of physical tools for manipulating virtual curves, their interaction techniques were limited by simple absolute mappings between physical tool and virtual curves/surfaces. Their system also provided little precision control over virtual curve parameters.

In this paper, we present a system that significantly extends this previous research, demonstrating the use of a high degree-of-freedom curve input device for quick but precise curve creation and manipulation in both 2D and 3D space. The system achieves this via a suite of new interaction techniques for *relative* mapping of the parameters of the physical device to the virtual world.

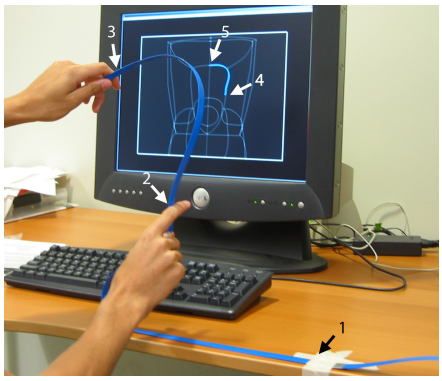

Fig. 1. System setup. The tape is secured at point #1. The first half of the tape (segment 1-2) is used to position and orient the starting point (#2) of the second half of the tape (segment 2-3). The position and shape of the second half of the tape (segment 2-3) is mapped to the virtual TapeWidget (segment 4-5).

CHI 2003, April 5–10, 2003, Ft. Lauderdale, Florida, USA.

SYSTEM HARDWARE

The primary input device that forms the core of our interface is the *ShapeTape* (www.measurand.com): a 96 x 1 x 0.1 cm rubber tape with a flexible spring steel core that has 32 fiber optic sensors distributed in pairs uniformly, 6 cm apart, along its length. Each sensor pair provides bend and twist information at its location, and by summing the bends and twists of the sensors along the tape, the shape of the tape can be reconstructed in three dimensions. This shape reconstruction is relative to the location of the first sensor pair. The tape was secured to a desk at the location of this first sensor pair. In [3], a separate position and orientation tracker was used to determine the location of the starting point of the tape in 3D space. Rather than adding another piece of hardware to our system, we instead used the first half of the tape to position and orient in physical 3D space the starting point of the second half of the tape. The second half of the tape was then used to input shape information. Figure1 illustrates this hardware setup.

The only other input device used was a footpedal hinged in the middle, with two momentary buttons: one at the front of the pedal (henceforth referred to as the *FrontButton*) and the other at the back (*BackButton*).

It is important to note that we have deliberately chosen this minimal hardware setup, in order see how far we could go with using only a curve input device and two buttons. As will be evident as we progress through this paper, we were able to develop a significant repertoire of gestural interaction techniques using only this minimalist configuration. We readily admit, however, that while this was an excellent setup for pushing the boundaries of our research, any commercially viable system for curve manipulation using curve input devices would likely require additional input modalities.

GESTURES

In building a usable system for curve manipulation using such a minimal hardware configuration, we are faced with the challenge of providing a mechanism for command input. We use the footpedal's two buttons for the most frequently used commands, and to kinesthetically maintain a few modes. Additional commands are specified via a set of gestures performed using the physical tape (henceforth referred to simply as *tape)*. Figure 2 illustrates this gesture set. Six of the gestures (Fig. 2a-f) are recognized by tracking the velocity vectors of the centre and two endpoint sensors of the tape. The last two gestures (Fig. 2g,h) are recognized by measuring the amount, direction, and timing of twisting of the tape. These eight gestures are used throughout our system. The commands associated with these gestures will be described as we progress through the paper explaining the various interaction techniques.

PHYSICAL to VIRTUAL INTERFACE

In the standard mouse/keyboard GUI interface, the system cursor serves as an abstract representation of mouse movements. This cursor is then used to manipulate various parts of the interface. In a sense, the cursor serves as an intermediary between the mouse and the rest of the interface. Unlike the standard point cursor that has only two changeable parameters (X-Y position in 2D space), analogous intermediaries for high degree-of-freedom devices would likely have more parameters that can be manipulated by the device. Based on previous experience [3], and building on foundational work on three dimensional widgets by Conner et. al. [6], we have designed such an intermediary for our interface, called the *TapeWidget.* Manipulations on the tape are used to control parameters of the TapeWidget, including position in space, size, and shape. The TapeWidget is then used to create, edit, and manipulate other virtual curves in the graphical scene. This TapeWidget is one major difference between this present system and the exploratory work done in [3].

The following subsections describe the various techniques we have developed that use manipulations of the tape to change the TapeWidget's parameters.

Tape to TapeWidget Mapping

Two points on the physical tape are mapped to the TapeWidget's endpoints (Fig. 1). Thus, the TapeWidget takes on the shape of this section of the physical tape.

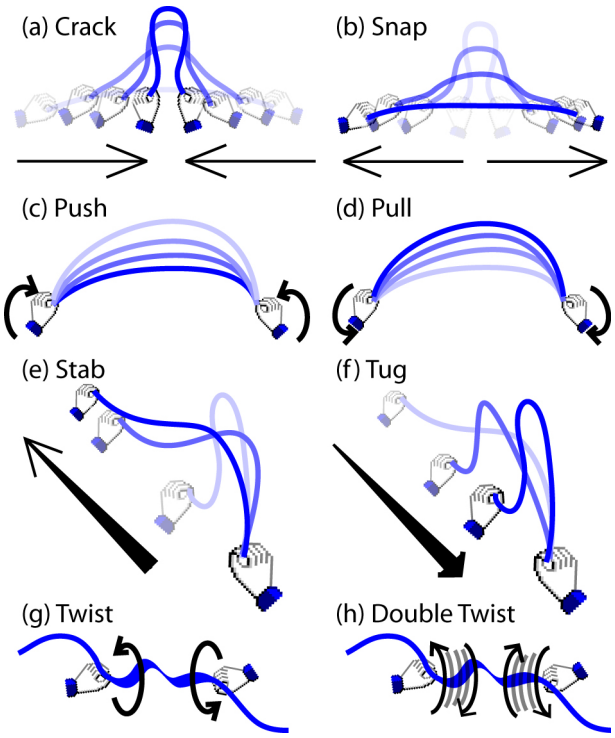

Fig. 2. Gesture set. (a) Crack – quickly move the tape endpoints close together. (b) Snap – quickly move the tape endpoints apart. (c) Push – quickly move the centre of the tape towards the screen by "flicking" the wrists forward. (d) Pull – quickly move the centre of the tape away form the screen by "flicking" the wrists backward. (e) Stab – quickly move the free endpoint of the tape towards the screen. (f) Tug – quickly pull the free endpoint of the tape away from the screen. (g) Twist – twist the tape with each hand moving in opposite directions. (h) Double Twist – twist the tape in one direction and then in the opposite direction in quick succession.

TapeWidget Positioning

The first half of the tape, which is used to track the location in space of the second half of the tape, does not allow for enough freedom to move the TapeWidget around the entire screen. As such, we need an interaction technique for gross scale position of the TapeWidget in space. This is accomplished as follows using a "flying" metaphor: when the FrontButton is pushed and held, moving the tape moves the TapeWidget in the same direction with a velocity relative to the tape's distance from it's starting point.

TapeWidget Scaling

Scaling the TapeWidget is accomplished by using a twist gesture on the tape. When the tape is twisted in one direction, the size of the TapeWidget is increased. A twist in the opposite direction scales down the TapeWidget. The twist gesture can be made anywhere along the tape.

Endpoint Mapping

The two endpoints that are set by default to map a section of the tape to the TapeWidget can be changed at any time using a double twist gesture. Double twisting at any point of the tape will make that the endpoint (Fig. 3) – analogous to the real-world action of twisting a piece of wire back and forth to break it. Together with scaling, this enables subsections of the tape to be mapped to the TapeWidget, resulting in changing the gain between the tape and the TapeWidget. If a small section of the tape is mapped to the entire TapeWidget, the resulting high gain mapping is good for changing the shape of the entire TapeWidget with just a small change in the shape of the tape. Conversely, mapping a large section of the tape to the TapeWidget results in a low gain mapping that is better for precise tweaking of portions of the TapeWidget. A snap gesture restores the default endpoint mapping.

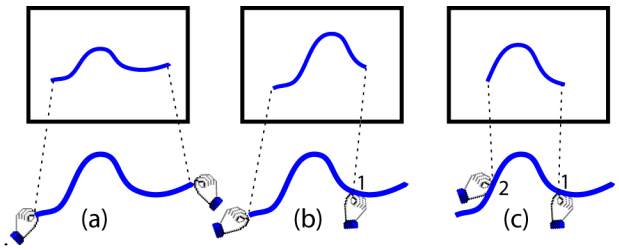

Fig. 3. Endpoint mapping. (a) shows the default mapping where the whole tape is mapped to the whole TapeWidget. (b, c) a double twist gesture at points 1 and 2 respectively sets those points as the new endpoints.

Sharp Corners

Since the tape cannot be physically bent into sharp corners (the fiber sensors would crack if bent too sharply), the TapeWidget's shape by default also cannot have sharp corners. However, in many curve editing tasks, it is desirable to be able to create sharp changes in a curve's shape. To support this, we use a crack gesture to "crack" the continuity of the TapeWidget's shape, resulting in a "corner" that consists of two straight lines joined at a vertex (Fig. 4). The location of the endpoints of this corner TapeWidget are controlled by the endpoints of the tape, and the angle between the lines is relative to the distance between the tape's endpoints. A snap gesture restores the regular curved TapeWidget.

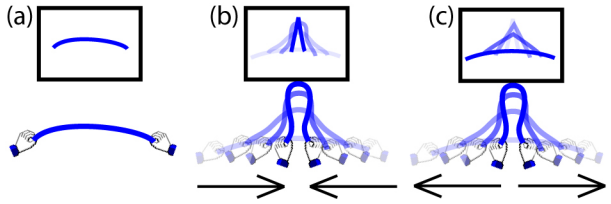

Fig. 4. Sharp corners. (a) default mapping of tape to TapeWidget. (b) a crack gesture changes the TapeWidget into a sharp corner. (c) snap gesture restores TapeWidget to default mapping in (a).

TapeWidget Locking

We have found many situations (to be described shortly) where it is desirable to temporarily lock the parameters of the TapeWidget. We use the BackButton as a toggle to lock and unlock the TapeWidget's position, shape, and size.

Relative Tape to TapeWidget Orientation Mapping

So far, apart from the technique for creating sharp corners, all the manipulations we have described result in the TapeWidget taking on the exact orientation of the tape. However, it is sometimes desirable to have a more relative mapping between the orientation of the tape and TapeWidget, particularly in situations where the desired orientation of the TapeWidget would otherwise necessitate holding the tape in an awkward position. To support relative orientation mapping, we first click the BackButton to lock the TapeWidget. The tape can now be reoriented in a comfortable pose by the user, without affecting the TapeWidget. When the BackButton is clicked again the TapeWidget is unlocked and its shape, position, and size responds to new manipulations of the tape, but with a transformed orientation (Fig. 5). Again, a snap gesture restores the default absolute mapping.

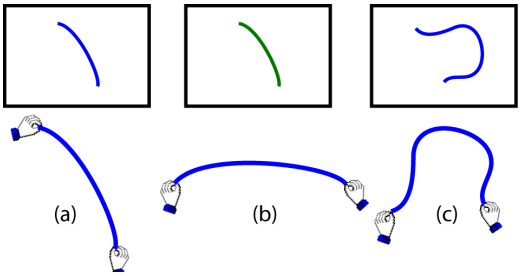

Fig. 5. Relative orientation mapping. (a) normal mapping. The tape is held in a fairly awkward position. (b). BackButton is clicked to lock the TapeWidget. The tape now be repositioned without affecting the TapeWidget. (c) BackButton is clicked again to unlock the TapeWidget. Tape manipulations are now mapped with a relative orientation to the TapeWidget.

These techniques result in a sophisticated interface between the physical tape and its virtual instantiation – the TapeWidget. The interface supports both simple and complex (but precise) control of the TapeWidget's parameters. Using this highly maneuverable TapeWidget, coupled with a few more gestures, we have developed a set

of interaction techniques for creating and manipulating other virtual curves in a graphical scene. We describe these interaction techniques in the following sections, beginning with 2D techniques, and then moving to 3D.

2D CURVE CREATION and MANIPULATION

Creation

Creating a new curve in the scene is accomplished by using a push gesture (Fig. 6). The metaphor here is that of "pushing forward to drop a curve onto the scene", echoing the paradigm shift from control point based creation to a faster, more direct approach. Invoking this gesture creates a new curve at the position of the TapeWidget that replicates its current shape and size. We lock the TapeWidget (using the BackButton) before invoking the push gesture. This prevents any movements in the tape caused by the gesture itself from accidentally changing TapeWidget parameters.

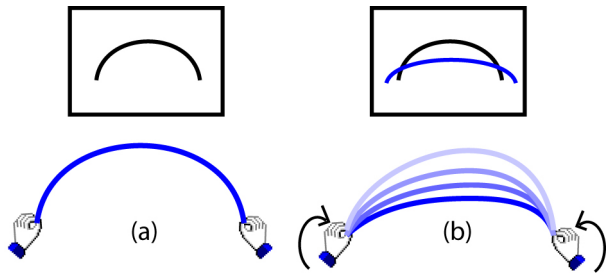

Fig. 6. Curve creation. (a) tape shapes and positions the TapeWidget. BackButton click locks the TapeWidget. (b) push gesture drops a new curve with TapeWidget's shape into scene.

Editing

Our curve editing design philosophy is based on a sculpting metaphor with the TapeWidget as the analogue of a sculpting tool. Just as a sculpting tool's behaviour changes depending on how it is used, we implicitly use the TapeWidget's proximity to, and intersection with, curves in the scene to determine the type of editing to be performed. We have developed four example editing techniques that as a whole can be viewed as a single editing mechanism that changes its behaviour depending on the proximity and intersection of the TapeWidget to the curve. The next few subsections describe these techniques.

Curve Selection

To edit a curve, the user must first select it. Our system considers the curve closest to the endpoint of the TapeWidget to be the "current curve". To distinguish the current curve from others in the scene, we render it as a thicker curve. Clicking the FrontButton toggles selection and deselection of the current curve. We can select multiple curves by moving the TapeWidget around the scene.

Reshaping with a Single Intersection Point

The simplest method for reshaping a curve is by intersecting one endpoint of the TapeWidget with a selected curve. Two interpolation endpoints are placed on either side of the point of intersection, at a default distance. A preview curve that is a Bezier interpolation joining these interpolation points and the endpoint of the TapeWidget is displayed in red, indicating to the user what the resulting change would look like (Fig. 7). Increasing or decreasing the interpolation interval is achieved by twisting the tape. Clicking the FrontButton results in the curve being reshaped as indicated by the red preview curve.

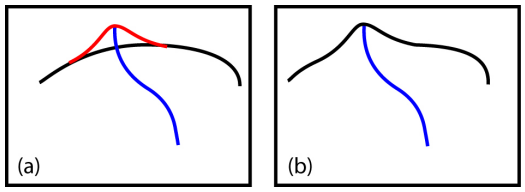

Fig. 7. Reshaping with a single intersection point. (a) the TapeWidget intersects a curve in the scene at one point, and the interpolated red preview curve is shown. (b) FrontButton is clicked and the curve takes on the shape of the preview curve.

Reshaping with Two Intersection Points

If the TapeWidget intersects a selected curve at two points, the red preview curves appears, taking on the shape of the TapeWidget between these two intersection points. The red preview curve is finely interpolated at the two points of intersection to maintain smoothness (Fig. 8). As with the previous technique, clicking the FrontButton results in the curve taking on the shape of the preview curve.

Both these methods of editing can be used while there is a relative orientation mapping between the tape and TapeWidget, and/or when the TapeWidget is shaped as a sharp corner as previously described.

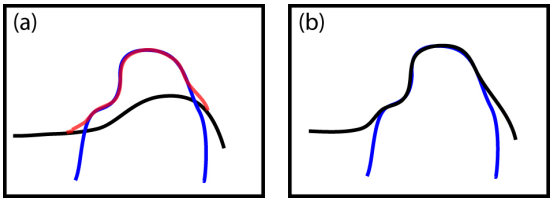

Fig. 8. Reshaping with two intersection points. (a) TapeWidget (blue) intersects a curve (black) at two points, and the resulting interpolated red preview curve is shown. (b) FrontButton is clicked and the curve takes on the shape of the preview curve.

Compound Reshaping

A third method, useful for precise but compound relative reshaping of curves, is now described with Figure 9 illustrating. First, the TapeWidget glues to an existing curve when its shape closely matches the shape of a curve near to it (Fig. 9a). This is the analogue of point snapping to parts of a scene when using a standard point cursor for editing. The endpoint extents are placed on the existing curve closest to where the TapeWidget endpoints were before the TapeWidget was glued (Fig. 9b). Once glued, the TapeWidget is controlled by both the endpoints and subsequent manipulation of the physical tape. When the TapeWidget's shape is changed, we keep the end points constrained to their original glued position by displacing every point on the TapeWidget by an offset vector as follows: The difference vectors between the original end points e_1, e_2, and the end points after the TapeWidget's shape is changed, are shown in Fig. 9c as d_1,d_2. The offset

vector at any point along the TapeWidget is an interpolation of the vectors d_1 and d_2, varying from d_1 at point e_1 to d_2 at point e_2. These offset vectors are added to the TapeWidget, with results as shown in Fig. 9d. Clicking the FrontButton reglues the TapeWidget in its current state, while preserving the same end points.

The other set of extents seen in Fig. 9b, called interpolation extents, are used to blend the results of the curve generated using the algorithm described above with the unedited segments of the curves. A bezier curve smoothly joins the unedited curve segments in the regions between the endpoints and the interpolation extents (see Fig. 9e). The end points and interpolation extents can also be edited from their default locations. Either set of extents are active at any given time. When the tape is twisted close to the center, the active set of extents move closer together or further apart, depending on the twist direction. If either endpoint of the tape is twisted, then only its corresponding extent will move. A double twist toggles the active set of extents.

A push gesture makes the existing curve permanently take on the shape of the preview curve in between the endpoint extents (Fig 9f). The TapeWidget remains glued, so the process can be repeated. As always, clicking the BackButton locks/unlocks the TapeWidget, allowing for the shape of the curve to be "cranked" in a relative manner (Fig. 10). This technique allows for precise, compound, relative reshaping of curves to be performed, which would be quite difficult to achieve using existing curve editing techniques. The TapeWidget unglues with a tug gesture.

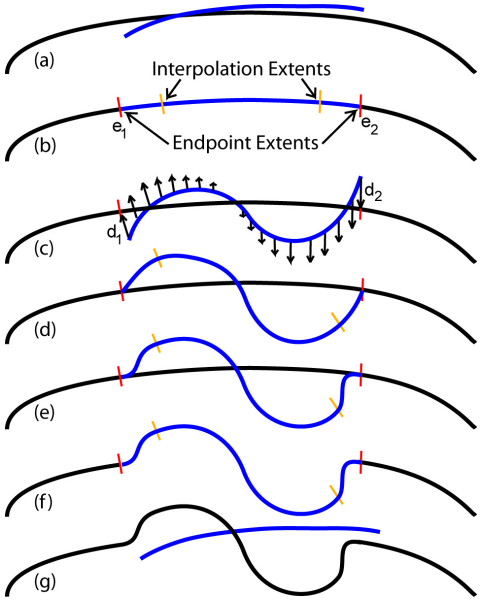

Fig. 9. Compound reshaping. (a) the TapeWidget (blue) moves close to an existing curve (black), causing it to “glue” onto the curve. (b) endpoint and interpolation extents are displayed. (c) the computed vector offsets (the arrows are for illustration only and do not appear when the system is in use). (d) shows result of adding the vector offsets. (e) curve is interpolated between the endpoint and interpolation extents. (f) a push gesture reshapes the curve. (g) a tug gesture unglues the TapeWidget from the curve.

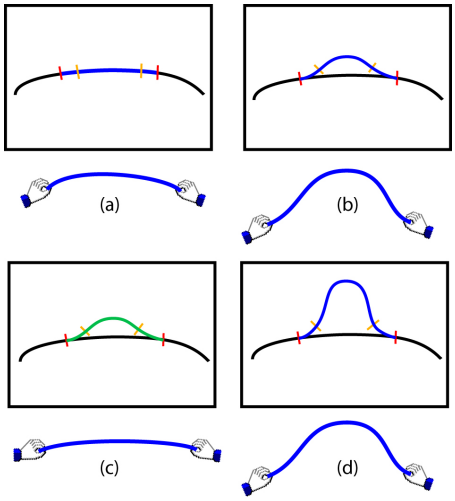

Fig. 10. Cranking during compound reshaping. (a) TapeWidget is glued onto the curve. (b) manipulating the TapeWidget shapes the curve. (c) BackButton is clicked to lock the TapeWidget, allowing the tape to be repositioned without affecting the TapeWidget. (d) BackButton is clicked again to unlock the TapeWidget. Manipulating the TapeWidget again results in the curve being further reshaped in a relative manner.

Extending Curves

If the TapeWidget is close to an endpoint of a curve, it glues to that endpoint. As usual, the TapeWidget can be locked and unlocked by clicking the BackButton. Twisting the tape determines how much the curve’s endpoint will be extended along the curved path provided by the TapeWidget’s shape (Fig. 11). A push gesture makes the curve extension permanent.

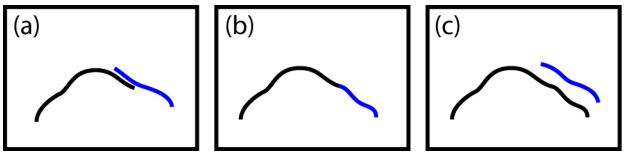

Fig. 11. Extending curves. (a) TapeWidget moves close to the endpoint of an existing curve. (b) TapeWidget glues to the endpoint of the curve. (c) after BackButton is clicked to lock the TapeWidget, a push gesture extends the curve.

Tools

When editing curves, it is often desirable to be able to reuse a previously defined TapeWidget shape. We support this by creating, saving, and recalling a set of user defined *tools*. To create a tool a crack gesture is made while the TapeWidget is locked. This closes the shape of the TapeWidget and locks it. The endpoint of the tape can then be used to control the position and rotation angle of the tool. The tool can be scaled and moved around the screen just like the regular TapeWidget. Similar to the TapeWidget, a tool can be used to drop new curves of the same shape as the tool into the scene, or to reshape existing curves. We also implemented a menuing system to provide the user with access to tools that have been previously created. A stab gesture pops up the menu directly above the position of the TapeWidget. The menu contains iconic

representations of the tools arranged in an arc (Fig. 12). A maximum of six tools are displayed at a time. If the menu has more than six tools, the edges of the menu fade out to indicate that the menu can be scrolled. Twisting the tape rotates the menu left or right, scrolling through all available tools. By moving the TapeWidget over a tool and clicking the FrontButton, the TapeWidget will take on the shape of that tool.

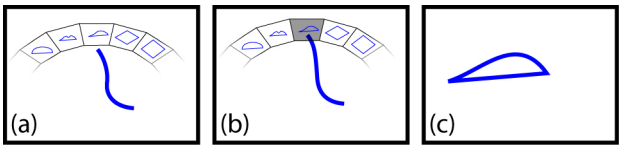

Fig. 12. Tool menu. (a) stab gesture pops up the menu. (b) moving TapeWidget highlights desired menu item. (c) clicking FrontButton selects highlighted tool.

If a tool is locked and a snap gesture is made, then the tool can be edited as if it were a curve in the scene. After modifications to the tool have been made, a tug gesture allows the user to return to using that tool. This new tool shape is also added to the tool menu.

3D OPERATIONS

We now extend our interactions into 3D space. Previous work [8, 9] on the design of 3D curve editing tools have indicated that to ensure accuracy it is preferable for 3D curves to be created and manipulated in 2D orthographic views of the 3D scene. Given that all the 2D techniques described in the previous sections will seamlessly work on 2D orthographic views of 3D space, we already have a suite of tools for 3D curve manipulation. What remains to be developed are techniques for use in the 3D perspective view, including: camera controls for maneuvering around the 3D view, selecting curves and construction planes, creating new 3D construction planes, and transitioning between 3D and 2D views.

2D to 3D Transitions

A tug gesture is used to seamlessly transition between 2D orthographic and 3D perspective views. Similar to the techniques used in [8, 9], this transition is smoothly animated to allow the user to understand the correspondence between the 2D and 3D visuals.

Camera Controls

As in previous systems [8, 9, 17], we support standard camera controls of tumble, pan, and zoom. However, we have adapted these techniques to work with the tape.

If the user points the tape towards the screen while pressing and holding down the FrontButton, movement of the tape's endpoint rotates the camera around the 3D scene.

If the tape is parallel to the screen while the FrontButton is pressed and held, a pan-zoom mode is entered. The two endpoints of the tape control the panning and zooming. Based on the technique used in [14], moving the two endpoints closer together or further apart zooms in or out respectively. Keeping the tape's endpoints at a constant distance apart and moving them together in the same direction pans the camera.

Construction Planes

In order to create curves in 3D space, we project 2D curves onto 3D construction planes. By default, three base construction planes (x-y, x-z, y-z), are drawn in the 3D scene. These not only provide a base for drawing curves onto, but also serve as a basis for specifying other construction planes.

Selecting Construction Planes

When in the 3D perspective view, a cursor is drawn and is controlled by the endpoint of the tape. Using this cursor to point at any of the construction planes and clicking with the BackButton selects that plane. Only one plane can be selected at a time.

Creating Curves on Construction Planes

Once a construction plane has been selected, a tug gesture smoothly transitions from the 3D perspective view to a 2D orthographic view perpendicular to that construction plane. In this 2D view, curves can be created and edited using all the techniques previously described. Any new curves created are projected onto the surface of the construction plane. Figure 13 illustrates. Another tug gesture returns to the 3D view.

Intersection Points.

In both 2D and 3D views, points where curves intersect planes are marked by a large green dot. This serves as a useful aid for the user to align curves.

Creating New Construction Planes

When an existing plane has been selected in the perspective view, a push gesture creates a new plane that's positioned perpendicular to this selected plane. The new plane is either flat or curved, depending on the type of plane last created. A snap gesture changes the plane from flat to curved, and a crack gesture does the opposite.

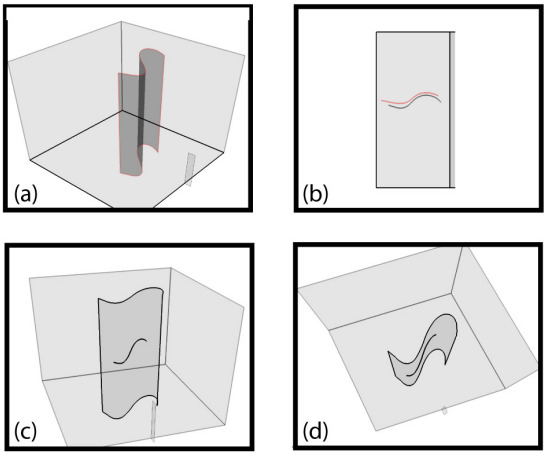

Fig. 13. Drawing on construction planes. (a) construction plane is selected. (b) tug gesture transitions to 2D orthographic view of that construction plane, and a new curve is created on it. (c, d) another tug gesture transitions back to the 3D perspective view, where the curve can be inspected in 3D space.

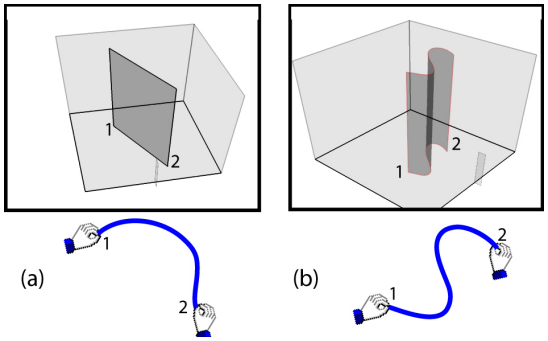

Fig. 14. Creating new construction planes. (a) in 3D view, a push gesture creates a flat plane. The endpoints 1 and 2 of the plane correspond to the endpoints of the tape that are similarly labeled. (b) a snap gesture converts to a curved plane whose shape is controlled by the tape.

As Fig. 14a shows, the location of a flat plane is controlled by the two endpoints of the tape. When the new flat plane's location is within a small delta of either major axis of the base selected plane, the flat plane will snap to that axis. Fig. 14b illustrates how the shape of a curved plane is determined by the shape of the tape.

A new plane can be moved around the 3D scene by moving the tape, using the same technique for moving the TapeWidget around when in the 2D orthographic view. The shape, position, and orientation of the plane can also be locked and unlocked by clicking the BackButton (i.e., using the same method for locking/unlocking the TapeWidget). When the new plane is locked, another push gesture confirms the addition of this plane, with its locked shape, into the scene at its current position and orientation.

The technique for plane creation described above works fine for planes of approximate shape. If more precise shape and location is required of a plane, its curvature can be defined in the orthographic view. First, a curve is drawn using the previously described techniques. With a pull gesture this curve will be used to define the curvature of a new surface, extruded along the normal of the currently selected plane.

USER FEEDBACK

While our system is still in the research prototype stage, we thought that it was important to get some early feedback from potential users. We felt that at this stage it would be more valuable to obtain feedback from expert users of other curve manipulation systems, rather than rely on novices who would be unlikely to understand all the subtleties involved in complex curve manipulation tasks. We asked two subjects very experienced in using various commercial 2D and 3D graphics software – one an academy-award nominated 3D modeler, and the other an industrial designer – to try out the system for a two hour session each. The first hour was used by the subject to learn the various gestures and interaction techniques. During the second hour, the subject was asked to freely use the system to create and manipulate curves of their own choosing.

Neither user chose to build complex 3D models with the system. One user did, however, build up a relatively simple 3D model of a table consisting of a circular body and four legs. Both users did become familiar enough with the system to get the overall feel of the various techniques, and were able to give us valuable feedback, leading to the following observations:

- Both users liked using the tape to directly manipulate curves without the abstractions found in current interfaces. However, they both felt that the tape would be more useful if it were complementing other tools and input devices, rather than being the *only* tool available. One user said he would like to be able to put the tape down and sketch part of a curve with a pen – in other words, using the best tool for the job as needed. Both users liked the fact that the tape could be manipulated using both hands simultaneously, and even suggested potentially new ways in which two-handed manipulation of the tape could be used.

- We found that the users were able to learn and perform our gesture set, and easily understood the underlying metaphors. In particular, the push gesture resonated with both users. This is likely because it only required a simple flick of the wrists and also because the metaphor was very obvious – that of pushing a curve or plane into the scene.

- One of the users commented on the amount of physical work that could be required to manipulate the tape. He said he much prefers a small tablet that can be used without lifting his wrist. We note that while there will always be some physical effort required when using tangible devices, more extensive use of the widget flying technique and relative mappings we provide could have significantly reduced the effort required. Almost all tangible user interfaces [7, 13] face this challenge of providing simple, easily understood physical artifacts to control virtual elements without increasing the work required of the user. Indeed, one of the reasons why the mouse is such a popular device is that it can be operated in a "lazy" fashion [1].

- One user mentioned that the paradigm which we used would be very useful for organic modeling, where curves are drawn and then tweaked to the designer's preferences. However, he said he sometimes felt uncomfortable creating and editing curves without direct control over the underlying mathematics of the curve. This complaint could be due to the fact that this user is very highly skilled in the use of current interfaces that demand that the user understand the underlying math. Indeed, one of the goals of our system was to insulate the user from the math! So, in a sense this user's comment could be viewed as a measure of how well we had managed to achieve this separation between the underlying math and the interaction techniques. On the other hand, this could be an argument for providing other tools to complement the tape so that the user would have the choice of either very direct manipulation with the tape or more abstract manipulations of the foundational curve parameters.

DISCUSSION, CONCLUSIONS, and FUTURE WORK

One of the first challenges we faced was finding a way to use the tape not only for the curve manipulation tasks that it was very well suited for, but also for command input. Our solution was to use the tape to capture user gestures, and we defined an initial set of eight distinct gestures. Note that each of these gestures was assigned a consistent meaning in our system. For example, the push gesture was consistently used to add elements to the scene: curves when in a 2D view, planes when in a 3D view. Similarly, the twist gesture was used to increase or decrease a particular variable: interpolation interval, widget size, and menu item position. Overall, we found that this small but well defined gesture set enabled us to support the fairly large set of interaction techniques used in our system.

While our minimalist hardware configuration was useful in forcing us to push the envelope on developing novel interaction techniques, including gestural command input, user feedback indicated that our system would benefit from integrating other techniques, such as sketching [4, 5, 10, 12, 17]. We also believe that it would be beneficial to have a suite of different physical tapes, each with unique physical characteristics such as bending tension, thickness, length, and precision. As discussed in [3], it could also be useful to have tapes that can maintain their physical shape over time.

An interesting design element of our system was the focus on a thorough set of 2D curve editing techniques, and then using these 2D techniques to create and manipulate 3D curves. Simply by adding construction planes and camera controls, we got a useful set of 3D tools "for free". However, given that we use 2D techniques to project curves onto 3D planes that are either flat or curved along one axis, only planar 3D curves are possible. In the current system, we restricted ourselves to using only the 2D shape information of the tape, although the tape itself can provide 3D shape data. While non-planar 3D curves alone are sufficient for creating many useful 3D wireframe models (Fig. 15), there are situations where it would be desirable to have non-planar 3D curves [9]. Accordingly, we intend to explore using the 3D shape information for creating non-planar 3D curves, building on techniques developed in [9].

Overall, our system has demonstrated how a high degree-of-freedom curve input device can be used for complex curve manipulations. It is also an example of a graspable [7] or tangible [13] interface that goes beyond the simple one-to-one mappings between physical and virtual artifacts that have typically been demonstrated by previous research.

ACKNOWLEDGEMENTS

We thank Lee Danisch of Measurand for providing the hardware, David Torre and Jeff Magder for video production, and members of the Dynamic Graphics Project laboratory (www.dgp.toronto.edu) at the University of Toronto for valuable comments and discussions.

VIDEOS

Digital video clips demonstrating this system can be downloaded from www.dgp.toronto.edu/~ravin/#videos

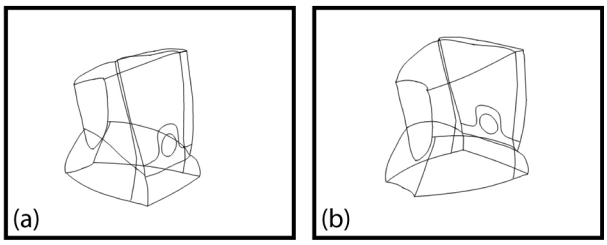

Fig. 15. Two views of an example wireframe model created entirely with our system.

REFERENCES

1. Balakrishnan, R., et. al. (1997). The Rockin'Mouse: Integral 3D manipulation on a plane. *CHI'97*. p. 311-318.
2. Balakrishnan, R., et. al. (1999). Digital tape drawing. *UIST'99*. p. 161-169.
3. Balakrishnan, R., et. al. (1999). Exploring interactive curve and surface manipulation using a bend and twist sensitive input strip. *ACM I3D'99*. p. 111-118.
4. Baudel, T. (1994). A mark-based interaction paradigm for free-hand drawing. *UIST'94*. 185-192.
5. Cohen, J., et. al. (1999). An interface for sketching 3D curves. *ACM I3D'99*. p. 17-21.
6. Conner, B., et. al. (1992). Three dimensional widgets. *Computer Graphics, 22*(4). p. 121-129.
7. Fitzmaurice, G., et. al. (1995). Bricks: Laying the foundations for graspable user interfaces. *CHI'95*. p. 442-449.
8. Grossman, T., et. al. (2001). Interaction techniques for 3D modeling on large displays. *ACM I3D'99*. p. 17-23.
9. Grossman, T., et. al.. (2002). Creating principal 3D curves with digital tape drawing. *CHI'02*.. p. 121-128.
10. Igarashi, T., & Hughes, J. (2001). A suggestive interface for 3D drawing. *UIST'01*. p. 173-181.
11. Igarashi, T., et. al. (1998). Path drawing for 3D walkthrough. *UIST'98*. p. 173-174.
12. Igarashi, T., et. al.. (1999). Teddy: a sketching interface for 3D freeform design. *SIGGRAPH'99*. p. 409-416.
13. Ishii, H., & Ullmer, B. (1997). Tangible bits: towards seamless interfaces between people, bits and atoms. *CHI'97*. p. 234-241.
14. Kurtenbach, G., et. al. (1997). The design of a GUI paradigm based on tablets, two-hands, and transparency. *CHI'97*. p. 35-42.
15. Sachs, E., et. al.. (1991). 3-draw: A tool for designing 3D shapes. *IEEE CG&A, 11*(6). p. 18-26.
16. Singh, K. (1999). Interactive curve design using french curves. *ACM I3D'99*. p. 23-30.
17. Zeleznik, R.C., et. al. (1996). SKETCH: An interface for sketching 3D scenes. *SIGGRAPH'96*. p. 163-170.